UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jan Šauli

PROTOTIP APLIKACIJE ZA PRIMERJAVO CEN ISTIH NEPREMIČNIN NA RAZLIČNIH NEPREMIČNINSKIH PORTALIH

DIPLOMSKO DELO

NA INTERDISCIPLINARNEM UNIVERZITETNEM ŠTUDIJU
RAČUNALNIŠTVA IN MATEMATIKE

MENTOR: doc. dr. Damjan Vavpotič

Ljubljana 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Prototip aplikacije za primerjavo cen istih nepremičnin na različnih nepremičninskih portalih

(*Prototype application for comparison of the price of the same real estate on different real estate portals*)

Tematika dela:

Na spletu obstaja vrsta nepremičninskih portalov, na katerih se pogosto pojavljajo oglasi za isto nepremičnino. Občasno se zgodi, da ima ista nepremičnina na različnih portalih različno ceno. Ker se opisi nepremičnine pogosto razlikujejo oz. so tudi v različnih jezikih, postane eden izmed najboljših možnih identifikatorjev fotografija nepremičnine. V okviru diplomske naloge preučite in predstavite informacijske tehnologije, ki bi jih lahko uporabili za iskanje iste nepremičnine po različnih portalih na podlagi fotografije in tehnologije za pridobitev osnovnih podatkov o nepremičnini s teh portalov – zlasti cene. Z uporabo preučenih tehnologij zasnujte delujoč prototip aplikacije, ki bo omogočal analizo in primerjavo cen istih nepremičnin na različnih nepremičninskih portalih. Predstavite delovanje in zgradbo prototipa. Na koncu tudi kratko predstavite rezultate analize.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Jan Šauli, z vpisno številko 63050380, sem avtor diplomskega dela z naslovom:

Prototip aplikacije za primerjavo cen istih nepremičnin na različnih nepremičninskih portalih

S svojim podpisom zagotavljam, da:

1. sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Damjan Vavpotič,

2. so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,

3. soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 4. septembera 2014 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Damjanu Vavpotiču za vso pomoč, usmerjanje, dostopnost in odzivnost pri izdelavi diplomske naloge.

Posebna zahvala gre staršem za podporo in spodbudo med šolanjem.

Zahvaljujem se tudi vsem prijateljem in bližnjim, ki so me do sedaj podpirali pri študiju in v življenju.

Kazalo

Seznam uporabljenih kratic

API (ang. *Application Programming Interface*) Programski vmesnik do specifične spletne storitve.

AJAX (ang. *Asynchronous JavaScript and XML*) Tehnika spletnega razvoja, uporabljena v prid asinhronim povezavam med odjemalcem in strežnikom.

DOM (ang. *Document Object Model*) Programski vmesnik za prikaz in delo z gradniki v dokumentih HTML, XHTML in XML.

RSS (ang. *Rich Site Summary*) Družina formatov spletnih virov, ki se uporabljajo na spletnih straneh z novicami ali blogih.

HTML (ang. *HyperText Markup Language*) Označevalni jezik, ki se uporablja pri opisu strukture spletne strani.

PHP (ang. *Hypertext Preprocessor*) Razširjen odprtokodni programski jezik, ki se uporablja za strežniške uporabe oziroma za razvoj dinamičnih spletnih vsebin.

W3C (ang. *World Wide Web Consortium*) Mednarodna organizacija, odgovorna za upravljanje s standardi na spletu.

OCR (ang. *Optical Character Recognation*) Mehanizirana pretvorba izrisanega teksta v elektronsko obliko.

URL (ang. *Uniform Resource Locator*) Enolični krajevnik vira je naslov spletnih strani v svetovnem spletu.

Povzetek

Zaradi trenutnih gospodarskih razmer je ponudba na nepremičninskem trgu dokaj pestra, vendar pa se prodajalci razmeram ne prilagajajo tudi s cenami. Kupec bi se lažje odločil za nakup nepremičnine, če bi bil informiran o trenutnem gibanju cen primerljivih nepremičnin in vzorcem prodaje, kot na primer za koliko in kdaj prodajalec spusti ceno.

Glavni cilj diplomske naloge je bil razviti prototip aplikacije za primerjavo cen nepremičnin na različnih nepremičninskih portalih. Aplikacija lušči nepremičninske oglase s pomočjo predlog iz različnih spletnih virov in jih nato medsebojno povezuje. Ključ za medsebojno povezovanje oglasov je slikovno iskanje. Aplikacija omogoča beleženje zgodovinskih sprememb in kasnejše dodajanje novih predlog.

V začetku diplomske naloge je teoretični del, kjer so opisani pojmi in koncepti, ki smo jih srečali in so bistveni za razvoj aplikacije. Osrednji del pa vsebuje opis delovanja in implementacije funkcionalnosti, ki jih izdelan prototip ponuja.

Ključne besede: PHP, ICDL, luščenje podatkov, slikovno iskanje, ontologije, nepremičnine, sematnični splet

Abstract

As per current economic situation the real estate market is very varied, but the sellers still do not change their prices to accomodate everchanging economic situation. It would be easier for buyer to decide to buy real estate, if he would be informed about current price fluctuation for comparable real estates and sales patterns.

Main focus of this thesis is to develop a prototype application for real estate prices comparison on different real estate portals. Application extracts real estate advertisements with the help of templates from different web portals and then links them through utilising reverse image search. Application allows logging changes to advertisements.

The thesis begins with theoretical part that describes terminology and concepts, which we encountered while constructing the prototype and are crucial for application development. Main part focues on the description of application operation and implementation of the functionalities that the prototype offers.

Keywords: PHP, ICDL, web crawling, reverse image search, ontology, real estate, semantic web

1 Uvod

Ker se v zadnjih časih vse več ljudi odloča za iskanje nepremičnin preko spleta, hkrati pa je vse več različnih spletnih portalov ki ponujajo oglase, smo se odločili raziskati, kako bi bilo možno le-te združiti v en sistem, kateri bi bil sposoben zaznati isti oglas na različnih portalih. S takim sistemom bi lahko preverili ali je cena določene nepremičnine enaka na vseh portalih oziroma ali se spreminja skozi čas.

Portalom ni v interesu, da bi svoje podatke delili, saj bi na tak način do njihovih podatkov lahko dostopali tudi konkurenčni portali, zato je verjetno to tudi glavni razlog, da se nihče ne odloča za deljenje podatkov preko programskih vmesnikov (API). Torej je edina možnost za dostop do podatkov preko luščenja vsebine iz HTML strani.

Drugi problem, ki ga srečamo pri pridobivanju podatkov iz različnih portalov, je v tem, da en oglas objavljen na različnih portalih težko identificiramo kot isti oglas. Najbolj učinkovit identifikator za določanje istih oglasov so slike v oglasih.

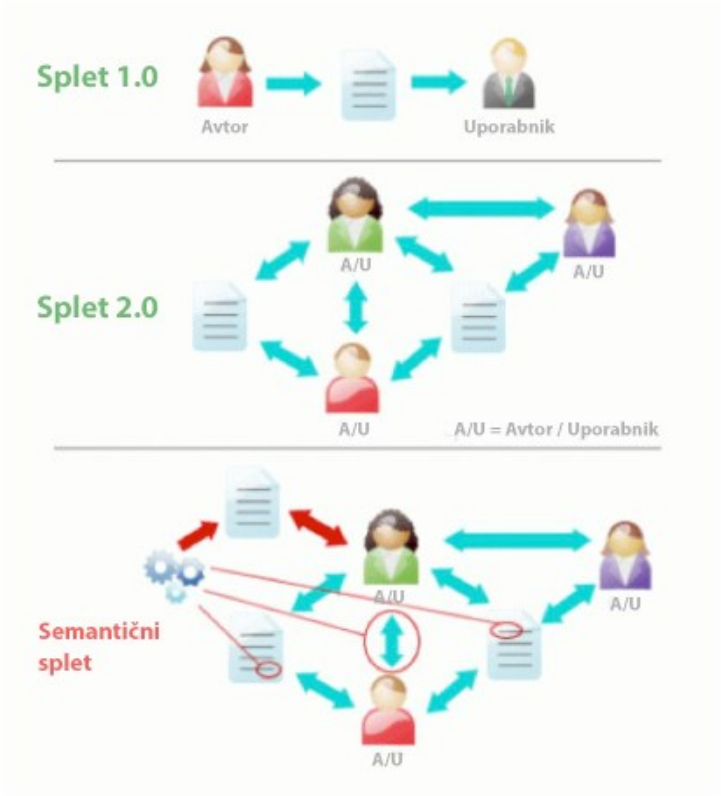
2 Osnovni pojmi

2.1 Spletni pajek (robot)

Spletni iskalniki uporabljajo algoritme, ki jim v slovenščini pravimo spletni pajki ali roboti, v angleščini pa so poznani pod izrazi *spider, boots, robots* ali *web scawler* [8]. Pajek je program, ki obišče spletne strani in bere iz njih, za namen posodabljanja svoje baze. Večina današnjih iskalnikov na spletu se poslužuje takšnih programov. Običajno so pajki programirani tako, da obiščejo spletne strani, ki so jih njihovi lastniki predložili kot nove ali posodobljene. Celotna spletna mesta ali posamezne strani lahko selektivno obišče in indeksira. Naziv pajki so dobili po tem, ker sočasno obiščejo veliko lokacij, s svojimi »nogami« pa so sposobni naenkrat pregledati velik del spleta. Po straneh spletnega mesta se lahko »plazijo« na več načinov. Najbolj pogost način je, da sledijo vsem hiperpovezavam na vsaki strani, dokler niso pregledali celotnega spletnega mesta.

2.2 Web 1.0, Web 2.0 in Web 3.0

Svetovni splet se počasi in vztrajno dopolnjuje in izboljšuje, kot je prikazano na sliki 1. Njegove začetke imenujemo kot Web 1.0, ki je uporabnike omejeval na pasivno gledanje vsebin. Nasledil ga je Web 2.0, kateri je še danes v veljavi. Glede na predhodnika omogoča veliko več interakcije. Nekateri pa že napovedujejo Web 3.0 oz. semantični splet, kjer naj bi bil splet boj razumljiv tudi za stroje in bi s pomočjo inteligentnih programov (agentov) analizirali podatke in jih nato ponudili uporabnikom.



Slika 1: Različice svetovnega spleta [22]

Web 1.0 je množica datotek, ne podatkov. Nestrukturirani dokumenti in statične strani so distribuirane po spletu, ki so medsebojno slabo povezane. Podatki so centralizirani. Z izjemo »pajkov«, ki potujejo po povezavah, je namenjen le ljudem, kar prikazuje slika 2. Imenujemo ga tudi *Read-only web*, saj je dovoljeval le iskanje in prebiranje. Kdaj točno se je zaključil Web 1.0 in začel Web 2.0 ne moremo točno določiti, ker se je ta sprememba zgodila postopoma skozi čas, ko je splet postajal vse bolj interaktiven. V splošnem se je to zgodilo med leti 1999 in 2004, ko je prišel širokopasovni internet, bolj izpopolnjeni brskalniki in AJAX.

Slika 2: Web 1.0 [6]

Web 2.0 ima začetke nekje v letu 2004, ko je prišlo do tehnoloških sprememb – nove internetne tehnologije. Web 2.0 omogoča uporabnikom interakcijo in sodelovanje drug z drugim, poleg tega pa še vključuje uporabnike kot aktivne člane, ki lahko sodelujejo pri ustvarjanju vsebin v virtualni skupnosti (slika 3). Ponuja veliko več kot je ponujal Web 1.0, ki je uporabnike omejil na pasivno gledanje vsebin, ki so bile ustvarjene zanje. Web 2.0 vključuje socialno mreženje, bloge, wikije, spletne aplikacije, itd [18]. Gre za splet, v katerem so podatki centralizirani. Do podatkov sicer ponekod lahko dostopamo s pomočjo tehnologij RSS in API, pomanjkljivost pa je še vedno zgolj spletišče posameznih dokumentov in aplikacij, ki centralizirane podatke hranijo samo zase. Poleg tega tudi ni dovolj razumljivo za stroje. To je razlog, da se je začelo razmišljati o nasledniku Web 3.0, ki bi bil razumljiv tudi strojem.

Slika 3: Web 2.0 [6]

Web 3.0 ni ločen splet, ampak je nadgradnja Web 2.0. Web 3.0 včasih imenujemo tudi semantični splet, čeprav obstajajo različne definicije. Eno od definicij je leta 2006 podal Tim Berners-Lee, ki je semantični splet opredelil kot sestavni del Web 3.0 [1]. Podatki so porazdeljeni, razdrobljeni, vseprisotni na spletu. To je splet, ki ni le za ljudi, ampak je razumljiv tudi inteligentnim strojem, ki znajo te podatke analizirati in ponuditi ljudem.

Web service ali spletna storitev je programska oprema sistema, ki podpira interakcijo med računalniki (stroji) preko interneta. Ponavadi so te storitve predstavljene kot API. Trenutno jih obstaja na tisoče in so za Web 3.0 bistvenega pomena, kajti ravno kombinacije semantičnih oznak in spletnih servisov lahko zagotovijo, da stroji komunicirajo med seboj neposredno.

Koncept semantičnega spleta je nastal v začetku 70-ih pod taktirko znanstvenika Allana M. Collinsa, jezikoslovca M. Rossa Quilliana in psihologinje Elizabete F. Loftus, ko so objavili različne članke o načinih predstavitve pomensko strukturiranih znanj [2][5][13][28]. To je bilo kasneje razširjeno na splet, kjer se je človeško berljivim hiperpovezavam začelo dodajati metapodatke. Metapodatki so bili berljivi tudi strojem. Tako so spletni agenti lahko bolj inteligentno dostopali do spleta. Izraz »Semantični splet« je izumil Tim Berners-Lee, direktor World Wide Web Consortium (W3C) in ga definiral kot: »Podatki spleta, katere lahko stroji posredno ali neposredno obdelujejo« [17].

Trenutno ima najvidnejšo vlogo pri pospeševanju uporabe semantičnega spleta prav mednarodna organizacija za standardizacijo W3C. Ta namreč spodbuja skupne formate podatkov na spletu. Leta 2004 je izdala 2 standarda: Resource Description Framework (RDF) in Web Ontology Language (OWL) [23].

2.3 Semantika in sintaksa

Princip dostopa do podatkov ne deluje več v striktnem iskanju, ampak na korelaciji med podatki in vsebino. Poudarek je predvsem na tem, da stroji ne pridobivajo podatkov na podlagi strukture, ampak na podlagi pomena. Kar je uporabniku preprosto, a zamudno opravilo, je trenutno velik izziv za računalnike [19]. V ta namen so bile razvite:

- Ontologije
- Meta podatki

2.3.1 Ontologije

Ontologija se šteje za enega od glavnih stebrov semantičnega spleta, vendar pa nima splošno priznane definicije [10].

Obstajata vsaj dve definiciji za ontologijo:

1. Ontologija je formalna specifikacija skupne konceptualizacije [27].
2. Glavna nit ontologije v filozofskem smislu je študija subjektov in njihovih odnosov. Vprašanji ontologije sta:
   -Katere vrste stvari obstajajo ali lahko obstajajo v svetu?
   -Kakšne odnose lahko te stvari imajo med seboj?
   Ontologija posveča manj pozornosti temu »kaj je« kot »kaj je mogoče« [11].

Slovarji ontologije vsebujejo znanja za organizacijo podatkov in se uporabljajo v različnih področjih: umetni inteligenci, semantičnem spletu, sistemskem inženiringu, razvoju programske opreme, biomedicinski informatiki, bibliotekarstvu …

Ne obstaja jasna razmejitev med tem kaj je »slovar« in kaj »ontologija«. Trend je, da se uporabi beseda »ontologija« za bolj zapletene in morda bolj formalno zbrane izraze, medtem ko se »slovar« uporablja tedaj, ko ni potreben tako strog formalizem [21].

Slovarji so definirani s pomočjo razredov, lastnosti ter odnosov med njimi. V praksi so slovarji lahko zelo zapleteni (z več tisoč izrazi) ali zelo preprosti (opisuje le enega ali dva koncepta).

2.3.2 Meta podatki

Meta podatki so podatki o podatkih. Meta podatki so se sprva uporabljali predvsem v knjižnicah. Ko so informacije postajale vedno bolj digitalne, so se meta podatki uporabljali tudi za opis digitalnih podatkov z uporabo standardnih meta podatkov za tisto disciplino. Z opisom vsebine in konteksta podatkovnih zbirk, se je kakovost izvirnih podatkov/datotek močno povečala. Na primer spletna stran lahko vključuje meta podatke, ki določajo v katerem jeziku je napisana, kje najti več informacij o tej temi in omogoča brskalnikom, da avtomatično izboljšujejo uporabnikovo izkušnjo. Glavni namen meta podatkov je lažje odkrivanje ustreznih informacij [9].

Meta podatki so v največji meri namenjeni strojem in aplikacijam, kateri bi sicer težko razbrali pomen iz vsebine. Meta podatki so podani na strukturiran način, ki je v večini primerov opredeljen z določenim standardom.

2.4 Luščenje podatkov

Običajno se prenos podatkov med programi doseže z uporabo podatkovnih struktur, ki so primerne za avtomatsko strojno obdelavo. Takšni formati in protokoli za izmenjavo imajo običajno vnaprej znano strukturo in so dobro dokumentirani, da je podatke možno enostavno razčlenjevati. Vendar pa je tak način prenosa namenjen strojem in ni berljiv ljudem.

Kadar moramo podatke predstaviti končnemu uporabniku, ne obstajajo pa protokoli za prenos podatkov, je uporaba luščenja edina možnost. Luščenje podatkov ignorira binarne podatke (kot so slike in multimedijski podatki), vizualni izgled, odvečne podatke in ostale informacije, ki so bodisi nepomembne ali pa ovirajo avtomatsko obdelavo.

Najpogosteje se luščenja podatkov poslužujemo kadar ne obstaja drug mehanizem, da bi povezali dva sistema, ki nimata združljive strojne opreme ali kadar ne obstaja primeren vmesnik, da bi se lahko povezali z nekim zunanjim sistemom. V slednjem primeru bo upravljalec sistema zaznal luščenje podatkov kot nezaželeno akcijo, zaradi večje obremenitve sistema.

Ta način na splošno velja za *ad-hoc*, neelegantno tehniko, ki se uporablja le kot zadnjo možnost, ko ni na voljo res nobenega drugega mehanizma. Pri tem postopku je potrebno veliko odvečnega procesiranja podatkov, poleg tega pa predstavitveni nivo namenjen interpretaciji končnemu uporabniku, pogosto spreminja svojo strukturo. Zaradi česar je potrebno nenehno dopolnjevanje in popravljanje, sicer bi pri luščenju prihajalo do napak.

2.4.1 Zaslonsko luščenje

Zaslonsko luščenje je postopek zbiranja podatkov iz zaslona ene aplikacije in prevajanje na način, da ga lahko prikazuje druga aplikacija. Ta tehnika se običajno uporablja za zajemanje podatkov iz starejših aplikacij in prikazovanje v modernejših uporabniških vmesnikih. Včasih se jo napačno zamenjuje s pojmom »vsebinsko luščenje«, kjer pa se na ročni ali avtomatski način uporablja za pridobivanje podatkov iz spletne strani brez dovoljenja lastnika[16].

Zaslonsko luščenje je edina možna rešitev ob posodobitvah v modernejši uporabniški vmesnik, kadar morda ni več na voljo izvorne kode, dokumentacije sistema, API vmesnikov ali programerja, ki bi poznal prejšnji, zastarel sistem.

Eno od prvih zaslonskih luščenj je bilo uporabljano že v devetdesetih letih prejšnjega stoletja, ko so ponudniki finančnih podatkovnih zbirk Reuters, Telerate, in Quotron prikazovali svoje podatke končnim uporabnikom v terminalni velikosti 24x80 znakov. Dostop do podatkov so imeli le registrirani uporabniki. Med njimi so bile tudi investicijske banke, ki so luščile podatke iz terminala in jih uporabljale v svojih podatkovnih bazah in finančnih izračunih. Omenjeni program za luščenje je bil vmesni člen in je nadomestil človeka, ki bi te podatke prepisoval iz terminala.

Zaslonsko luščenje se uporablja tudi pri optični prepoznavi znakov (OCR). OCR je elektronska pretvorba skeniranih ali fotografiranih slik, ki vsebujejo besedilo, v računalniško berljiva besedila. Najpogosteje se uporablja kot način vnosa podatkov iz papirnatih dokumentov, kot so potni listi, računi, bančni izpiski, potrdila, vizitke, poštne pošiljke, ali drug poljubni tiskan zapis [12]. To je običajna metoda digitalizacije tiskanih besedil, da se jih lahko elektronsko ureja, išče, preverja, bolj kompaktno skladišči, objavlja na spletu, ali se jih uporablja v strojnih procesih, kot je strojno prevajanje. OCR spada v področje prepoznavanja oblik, umetne inteligence in računalniškega vida [4].

2.4.2 Spletno luščenje

Spletno luščenje je programska rešitev pridobivanja informacij iz spletnih strani. Običajno takšni programi simulirajo končnega uporabnika in raziskujejo po svetovnem spletu. To lahko izvedejo na dva načina, bodisi z izvajanjem na nižjem nivoju preko protokola HTTP, ali na nivoju spletnega brskalnika, kot je Internet Explorer ali Mozilla Firefox.

Spletno luščenje je tesno povezano s spletnim indeksiranjem, kjer se spletni agenti ali spletni pajki sprehajajo po spletnih straneh in iščejo informacije za indeksiranje. To je univerzalna tehnika večine spletnih iskalnikov. V nasprotju s spletnim indeksiranjem se spletno luščenje bolj osredotoča na preoblikovanje nestrukturiranih podatkov na spletu, običajno v HTML obliki, v strukturirane podatke, ki se shranjujejo v lokalno podatkovno bazo in nato analizirajo.

Spletno luščenje lahko uporabljamo za različne namene, kot so primerjave cen, luščenje kontaktov, spremljanje podatkov o vremenu, odkrivanju sprememb na spletnih straneh, raziskovanju … Podjetja, kot so Amazon AWS, Webdatacapture.com, 80legs.com in Google zagotavljajo končnim uporabnikom orodja za spletno luščenje brezplačno.

Poročilo, ki temelji na podatkih ene največje podatkovnih baz na svetu za spletno luščenje poroča, da se je promet povezan s spletnim luščenjem v zadnjih letih močno povečal. V povprečju je bilo 23% vsega spletnega prometa v letu 2013 povezano prav s spletnim luščenjem [15].

2.4.2.1 Tehnike spletnega luščenja

Spletno luščenje je proces samodejnega zbiranja informacij iz svetovnega spleta. Obstajajo različne stopnje avtomatizacije za spletno luščenje [24]:

- **Ročno kopiranje** (ang. *copy-and-paste*): Včasih celo najboljši programi za spletno luščenje ne morejo nadomestiti človeškega ročnega kopiranja in je lahko to edina rešitev, ko so spletne strani izdelane po principih, ki omejujejo strojno spletno luščenje.
- **Regularni izrazi**: Preprost, a močan pristop za pridobivanje informacij s spletnih strani lahko temelji na regularnih izrazih programskih jezikov kot sta na primer Perl ali Python.
- **HTTP programiranje:** Vsebino statične in dinamične spletne strani se lahko pridobi s pošiljanjem HTTP zahtevka na oddaljeni strežnik s programiranjem vtičnice (ang. *socket*).
- **HTML razčlenjevalniki:** Mnoge spletne strani imajo dinamično generirane strani na podlagi strukturiranega vira (npr. podatkovne baze). Podatki v isti

kategoriji so kodirani na podlagi istih/podobnih strani z istimi predlogami. Največkrat velja za take skupine strani, da imajo zelo podobno strukturo URL naslova in jih je zaradi tega enostavno identificirati. Pri podatkovnem luščenju program, ki zazna take predloge znotraj določenega vira informacij, izvleče njeno vsebino in jo prevede v relacijski obliko.

- **DOM razčlenjevalniki:** z vgradnjo spletnega brskalnika kot na primer Internet Explorer ali Mozilla, ki vračajo dinamično vsebino, ki je delno generirana tudi na strani odjemalca, spletno stran razčlenijo v DOM drevo, na podlagi katerega lahko programi pridobijo samo dele spletne strani in jih nato obdelajo.
- **Programska oprema za spletno luščenje:** Na voljo je veliko programskih orodij, katera ponujajo luščenje podatkov po meri. Ta programska oprema lahko poskusi samodejno prepoznati podatkovno strukturo strani ali pa omogoča snemanje uporabniškega vmesnik. Ta dva principa odpravljata potrebo po ročnem pisanju kode ali skript za luščenje, ki jih je mogoče uporabiti za pridobivanje in preoblikovanje vsebine.
- **Vertikalno združevanje platform:** Obstaja nekaj podjetij, ki so razvile posebno platformo za vertikalno luščenje. Te platforme so ustvarile kopico spletnih »pajkov« za posebno vertikalno preiskovanje ciljnih strani. Priprava vključuje vzpostavitev baze znanja za celotno vertikalo in nato platforma samodejno ustvarja pajke. Platforma je robustna in zagotavlja pridobivanje kakovostnih podatkov, hkrati pa je razširljiva (saj se lahko hitro razširi še na dodatna spletna mesta). Ta prilagodljivost je zelo koristna za spletna mesta z veliko podstranmi, kjer je iskanje podatkov zapleteno.
- **Računalniški vid za analizo spletnih strani:** Obstajajo prizadevanja, da bi s pomočjo strojnega učenja in računalniškega vida identificirali in pridobivali informacije iz spletnih strani enako kot to počnemo ljudje.

2.5 Slikovno iskanje

Slikovno iskanje se uporablja za iskanje enakih oz. podobnih slik po spletu. Kadar iščemo po sliki, na primer nepremičnine, lahko najdemo spletne strani kjer se ta slika nahaja, spletne strani kjer so podobne slike ali spletne strani, ki o teh nepremičninah le govorijo oziroma jih navajajo.

Takšno iskanje je najbolj učinkovito kadar iščemo po sliki ki je preprosta in se velikokrat pojavlja na spletu. Zato bo več zadetkov v primeru kadar iščemo po slikah znamenitosti, kot po osebnih slikah, kot so na primer družinske slike.

Ko iščemo po sliki, rezultati lahko vsebujejo:

- slike, ki so podobne naši sliki
- spletne strani, ki vsebujejo ujemajočo sliko

Slikovno iskanje nam ponuja kar nekaj ponudnikov [7]:

- Google (http://images.google.com/)
- TinEye (https://www.tineye.com/)
- Yandex (https://yandex.com/images/)
- Yahoo (http://images.search.yahoo.com/)
- Bing (https://www.bing.com/?scope=images&FORM=Z9LH1)
- Picsearch (http://www.picsearch.com/)
- Flickr (https://www.flickr.com/)
- …

Večina ponudnikov podpira samo iskanje slik glede na ključne besede, nekateri pa podpirajo tudi slikovno iskanje. Slednji so: Google, Yandex in TinyEye. Vsi podpirajo enake načine iskanja: iskanje po ključnih besedah, po URL povezavi, *drag-drop* slike in *upload* slike. V večini primerov pa Google za isto poizvedbo vrne več rezultatov kot preostala dva. Google vse slike po katerih iščemo (*upload* ali URL) shrani in, po navedbah Google-a, uporablja izključno samo zaradi izboljšanja svojih storitev [14].

2.5.1 Uporaba iskanja slik po vsebini v praksi

Slikovno iskanje je ena od priročnih inovacij, pri kateri pa je težko določiti točno uporabo. Iskanja se poslužujemo z namenom detektiranja originalnih slik, lahko pa je uporabno tudi za iskanje imena izdelka, nepremičnin ali na primer vizualizacije receptov.

Sprva je bilo namenjeno slikovno iskanje predvsem za:

- Uporabnike, ki so želeli poiskati podobne slike
- Uporabnike, ki so želeli poiskati originalen vir slike
- Uporabnike, ki so preverjali ali je kdo uporabil njihove osebne slike na spletu
- Fotografe in imetnike pravic, ki so iskali spletne strani, ki uporabljajo njihove slike brez dovoljenja

Po navedbi portala lifehacker.com [3] uporabniki uporabljajo slikovno iskanje za različne namene:

1. **Iskanje nepremičnin**

   Na prvi pogled je slikovno iskanje neuporabno za iskanje nepremičnin. Vendar je v nekaterih primerih možno priti do podatkov, ki bi sicer bili plačljivi. Obstajajo namreč spletne strani, ki imajo ekskluzivne oglase stanovanj, omogočajo brezplačno iskanje po svojih oglasih, vendar pa so kontaktni podatki na voljo samo uporabnikom, ki so plačali članarino. S pomočjo preiskovanja po sliki je možno najti še vse ostale strani, kjer se nahaja omenjena nepremičnina, in je tako verjetnost da pridemo brezplačno do kontaktnih podatkov kar velika.

2. **Iskanje imen neoznačenih izdelkov**

   Uporabniki, ki pogosto preiskujejo spletna mesta kot so Pinterest ali Tumbir, pogosto vidijo zanimive izdelke, ki pa niso pravilno označeni. Za nadaljnje poizvedovanje bi morali ugotoviti vsaj ime izdelka. Lahko gre za nekaj preprostega kot je poročna obleka ali pa za kompleksno kot je slika sobe z zanimivim stolom. Kadar gre za iskanje slike iz kataloga je ta metoda skoraj vedno uspešna, je pa pogosto koristna tudi v primerih kadar iščemo neznan izdelek na posnetku, ki je obdan z množico ostalih predmetov.

3. **Iskanje receptov na podlagi slike**

   Včasih naletimo na sliko jedi, ki je videti okusna, vendar pa ni poimenovana. Če poskusimo poiskati po omenjeni sliki, obstaja velika verjetnost, da najdemo recept jedi, če je bila le-ta uporabljena na kakem blogu ali na kater od spletnih strani z recepti.

4. **Iskanje imen znanih osebnosti**

   Če poskušamo prepoznati slavno osebo in nimamo veliko informacij, lahko z iskanjem po vsebini slike hitro pridobimo množico informacij, ki jih iščemo.

5. **Odkrivanje lažnih profilov in objav na socialnih omrežjih**

   Na večini socialnih omrežjih obstaja mnogo lažnih profilov, ki nas dodajajo za prijatelje. Ker takšne osebe pogosto za svoj profil uporabljajo slike, ki so naključne slike iz spleta, s tem postopkom lažnih profilov ni težko prepoznati.

   Na enak način lahko ugotavljamo tudi lažne objave uporabnikov. Uporabniki na socialnih omrežjih objavljajo na svojih profilih novice, ki

lahko vsebujejo slike. Z preiskovanjem po slikah lahko takšne nadležne dezinformacije dokaj enostavno prepoznamo.

6. **Iskanje ozadji v večji ločljivosti**
   Ena od bolj razširjenih uporab iskanja po vsebini slik je za iskanje originalnih slik v večji ločljivosti.

7. **Iskanje nedovoljene uporabe lastnih slik**
   Fotografi in umetniki, ki svoje fotografije oziroma umetnine objavljajo na spletu, želijo vedeti, če je njihovo slika uporabljena morda še kje na spletu in ob tem niso upoštevane avtorske pravice.

8. **Iskanje lastnikov**
   Če iščemo prvotnega lastnika slike, ki smo jo našli na spletu je slikovno iskanje skoraj vedno uspešno. Vendar ne gre samo za slike na spletu, možno je najti avtorja tudi za umetniške slike, karikature, strip, fotografije oziroma karkoli kar najdemo v vsakdanjem svetu.

9. **Prepoznavo rastlin, živali …**
   Če poskušamo ugotoviti ime rastline ali živali, ki jo najdemo v vsakdanjem življenju, jo lahko fotografiramo nato pa s pomočjo fotografije ugotovimo več o njej. Slikovno iskanje je priročno za identifikacijo vseh vrst živih bitij, čeprav so rezultati nekoliko manj predvidljivi kot tisti, ki so navedeni zgoraj.

3 Zasnova in izdelava prototipa

Prototip (v nadaljevanju aplikacija) smo želeli narediti s tehnologijami, ki so brezplačne in hkrati delujejo na vseh sistemih. To je bil razlog, da smo aplikacijo razvijali v jeziku PHP in podatke shranjevali v podatkovno bazo MySql. Poleg tega večino strežnikov omogoča avtomatsko zaganjanje skript kot je PHP s pomočjo razporejevalnikov (ang. *job scheduler* oz. *cron job*).

Med razvojem smo opazili, da je jezik PHP za našo nalogo nekoliko okoren. Do težav je prihajalo na določenih spletnih straneh zaradi drugačne predstave vsebine dokumenta HTML, kot ga vidi uporabnik in vsebine, ki jo pridobimo s pomočjo jezika PHP. Jezik PHP se izvaja na strani strežnika (ang. *server-side*), to pa je razlog, da ignorira dinamično spreminjanje in dodajanje na strani. Da bi se izognili omenjeni težavi, bi morali vključiti PHP modul, ki bi bil zmožen prikazati vsebino strani na enak način kot to počnejo spletni brskalniki. Med vsemi preizkušenimi je bil še najbolj primeren in enostaven za uporabo *Ultimate Web Scraper Toolkit* [20], ki pa tudi ni popolno zadostil našim zahtevam. Spoznali smo, da na enak način kot brskalniki podpira le preusmeritve in uporabo piškotkov, ne podpira pa prikaza v celoti. Tako smo se odločili uporabiti knjižnico cURL, ki že vsebuje ostale funkcionalnosti, ki so nam potrebne. Glede na to, da predloge za luščenje spletnih strani nastavljajo administratorji, ki pa naj bi poznali strukturo HTML, ni težave v kolikor je vsebina nekoliko drugačna kot jo prikazujejo spletni brskalniki. Z metodo luščenja podatkov iz 20-ih različnih spletnih strani nam je uspelo pridobiti vse podatke, ki smo jih potrebovali. V večini primerov so bile najbolj problematične galerije slik, ki se izrisujejo dinamično, vendar smo s preučevanjem strukture spletne strani uspeli izluščiti tudi te dinamično generirane dele strani.

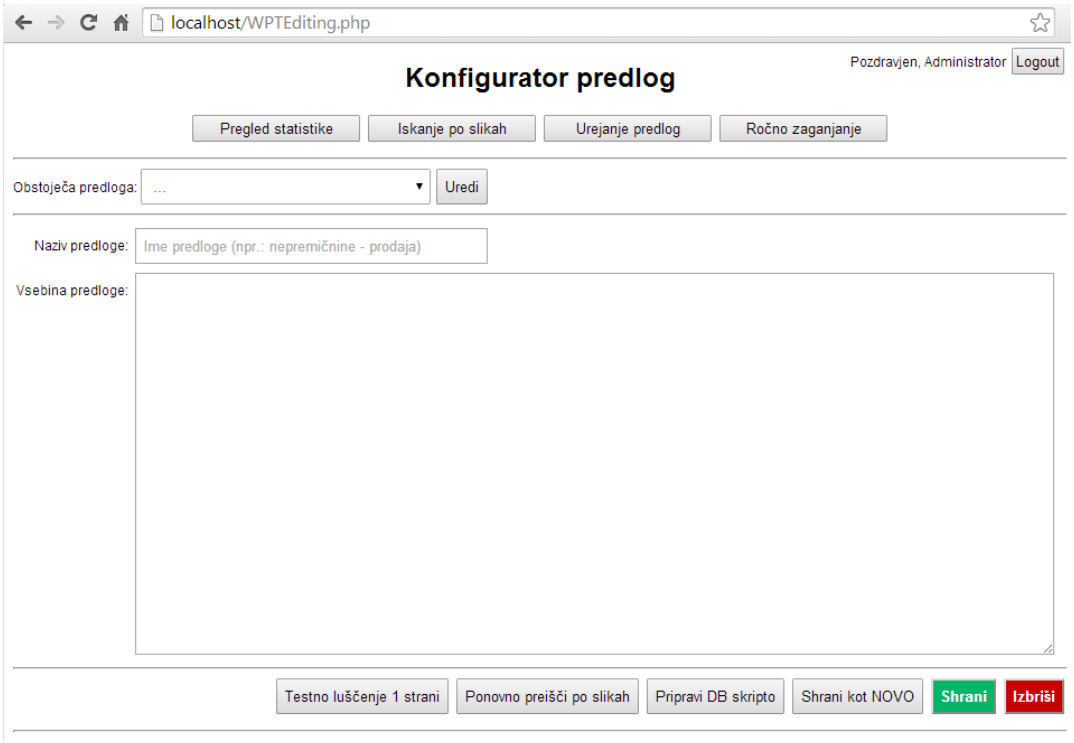
3.1 Delovanje aplikacije

Z vidika možnih zlorab, dostop do takšne aplikacije ne bi smel imeti kdorkoli in ima zato tudi naša aplikacija prijavo uporabnikov. Smiselno je, da lahko aplikacijo uporablja samo skrbnik sistema, ki ima znanje za pripravo predloge. Registracija novega uporabnika niti ni dovoljena iz že prej podanega razloga. V kolikor obstaja potreba po dodajanju novega uporabnika, ga lahko doda le obstoječi uporabnik.

3.1.1 Urejanje predlog

Slika 4 prikazuje osnovno okno aplikacije, v katerem izvedemo začetne nastavitve in nam omogoča dodajanje in urejanje predlog za luščenje. Preden predlogo shranimo v bazo lahko testno poženemo luščenje, da preverimo če predloga deluje kot smo želeli. Luščila se bo samo prva spletna stran, ki bo ustrezala predlogi. Podatki pridobljeni iz testnega luščenja se ne

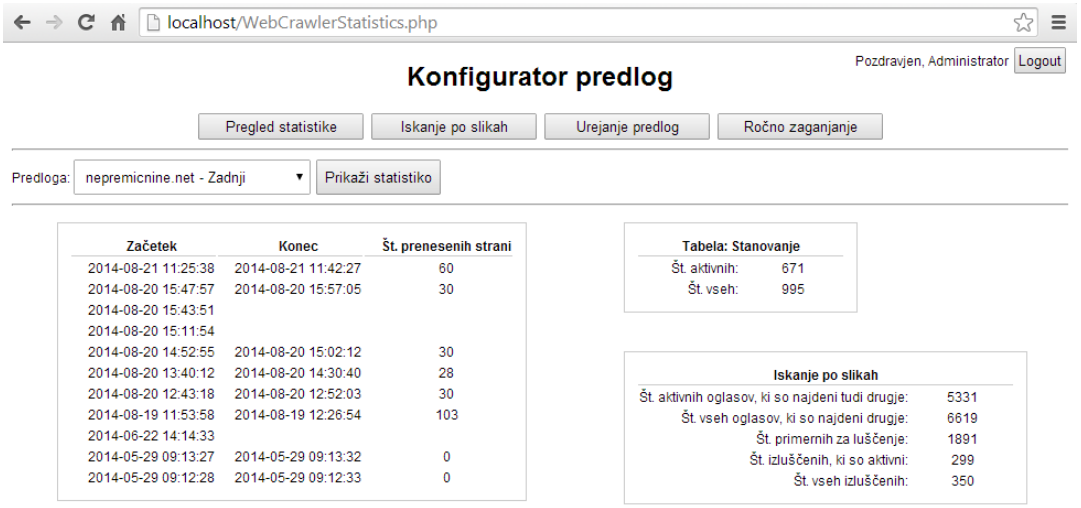
shranijo v bazo, ampak se samo prikažejo na strani. V kolikor je predloga primerna za shranjevanje, jo lahko shranimo kot obstoječi zapis ali nov zapis. Priporočljivo je da spremenimo tudi naziv predloge, kadar jo shranjujemo kot nov zapis. Obstaja tudi možnost brisanja predlog. Za izbrano predlogo nam aplikacija lahko pripravi skripto MySql za generiranje potrebnih tabel in prožilcev, da si lahko izluščene podatke shranimo v svojo podatkovno bazo.



Slika 4: Ekranska slika Konfigurator predlog

3.1.2 Pregled statistike luščenja

Za vsako predlogo imamo možnost vpogleda v njeno statistiko. Iz statistike dobimo podatke za zadnjih 10 luščenj; kdaj so bila zagnana, kdaj končana in koliko oglasov je bilo izluščenih. Poleg tega imamo informacijo o skupnem številu izluščenih podatkov (slika 5).

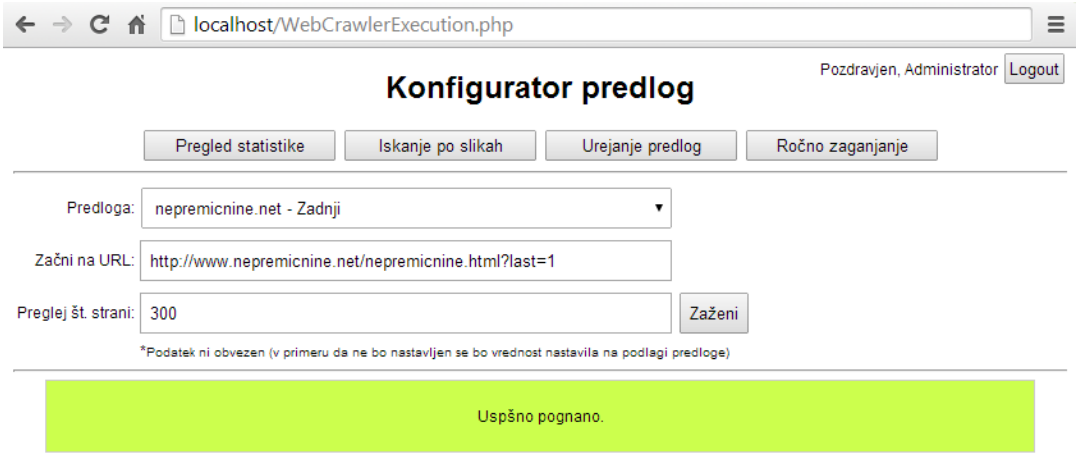


Slika 5: Ekranska slika pregleda statistike luščenja

3.1.3 Ročno zaganjenje luščenja

Luščenje lahko zaženemo ročno. Kot je iz slike 6 razvidno, je potrebno nastaviti parametra iz katere predloge želimo luščiti in število strani, ki naj bodo pregledane.

Omogočena je tudi nastavitev URL naslova, za katerega želimo, da se prične z preiskovanjem. V kolikor se vrednost ne nastavi, je privzeta vrednost iz predloge.



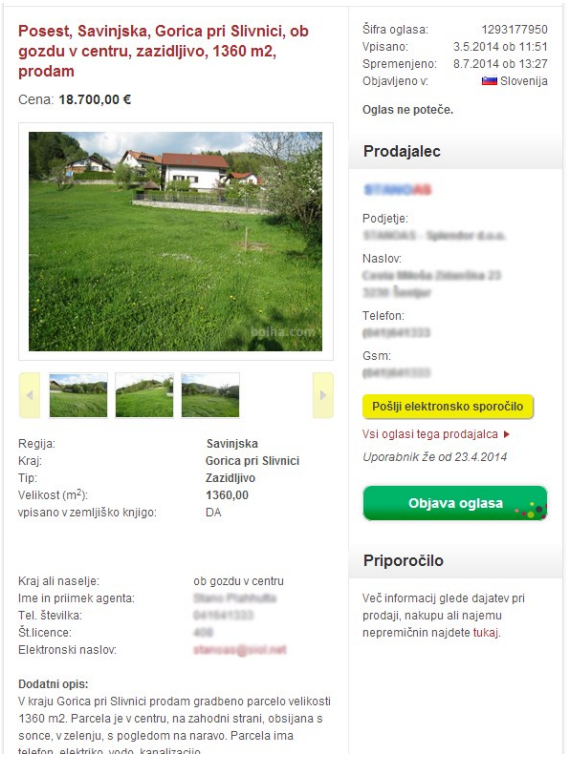
Slika 6: Ekranska slika pregleda ročnega zaganjanja luščenja

Poleg ročnega zaganjanja, sta na voljo še dva načina zaganjanja, ki pa ju ni mogoče pognati preko aplikacije, zato se ju poganja avtomatično preko razporejevalnikov. Prvi način je namenjen iskanju novih oglasov, drugi pa posodabljanju že izluščenih. Pri iskanju novih

oglasov se poišče predlogo, ki najdlje časa ni bila zagnana in se z njo poskuša najti nove oglase. Pri posodabljanju zapisov pa se poskuša izluščiti tudi oglase, ki so bili najdeni na podlagi slik. Pri iskanju novih oglasov se za omenjene zapise le polnijo informacije, kje se slike nahajajo, medtem ko se takrat ne lušči podatkov, ki so najdeni glede na slike.

3.1.4 Pregledovanje zapisov pridobljenih z iskanjem po slikah

Slikovno iskanje ima pomembno vlogo v naši aplikaciji. Omenjena funkcionalnost ima veliko dodano vrednost in zelo razširjene možne namene uporabe, ki smo jih opisali v poglavju 2.5. Vendar pa smo med testiranjem ugotovili, da občasno dobimo napačne rezultate za poizvedbo. Kot primer lahko pogledamo oglas na Bolha.com (slika 7):

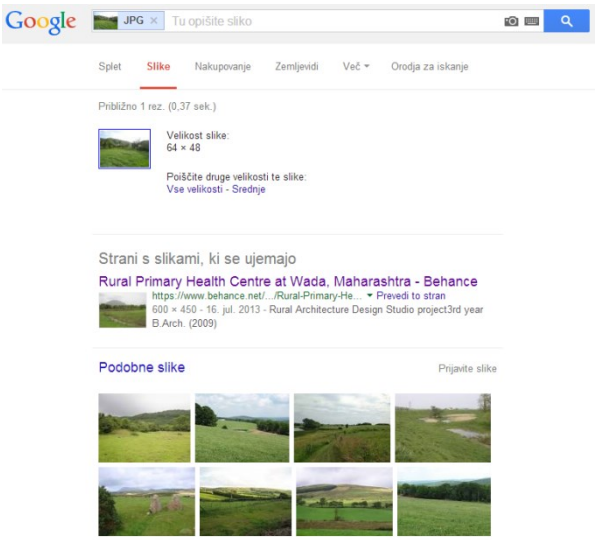


Slika 7: Oglas na Bolha.com

Za spodnjo, sliko 8, želimo poiskati spletne strani s slikami, ki se ujemajo:

Slika 8: Primer slike za iskanje

Google nam vrne samo en primeren zadetek (slika 9) za iskano sliko.

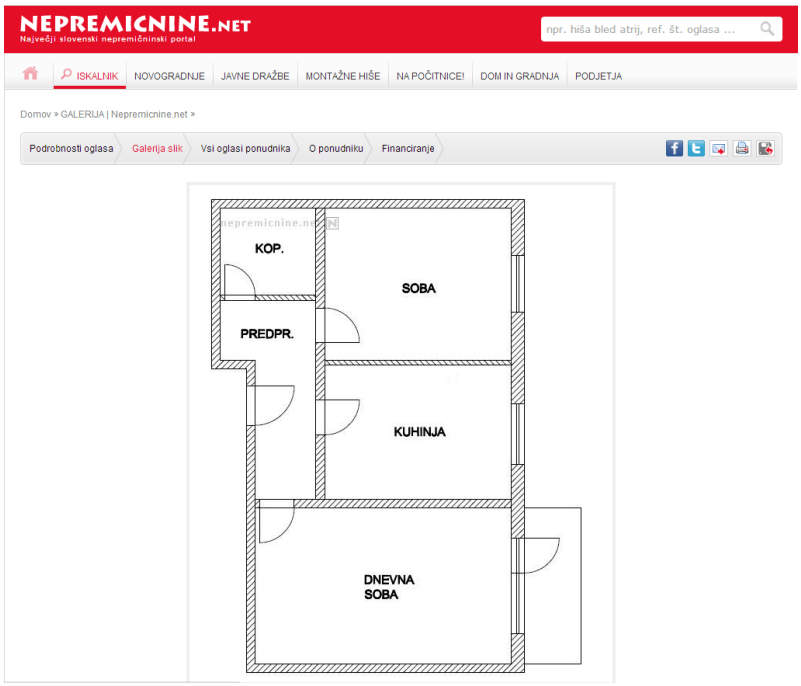


Slika 9: Rezultat iskanja z Googlom

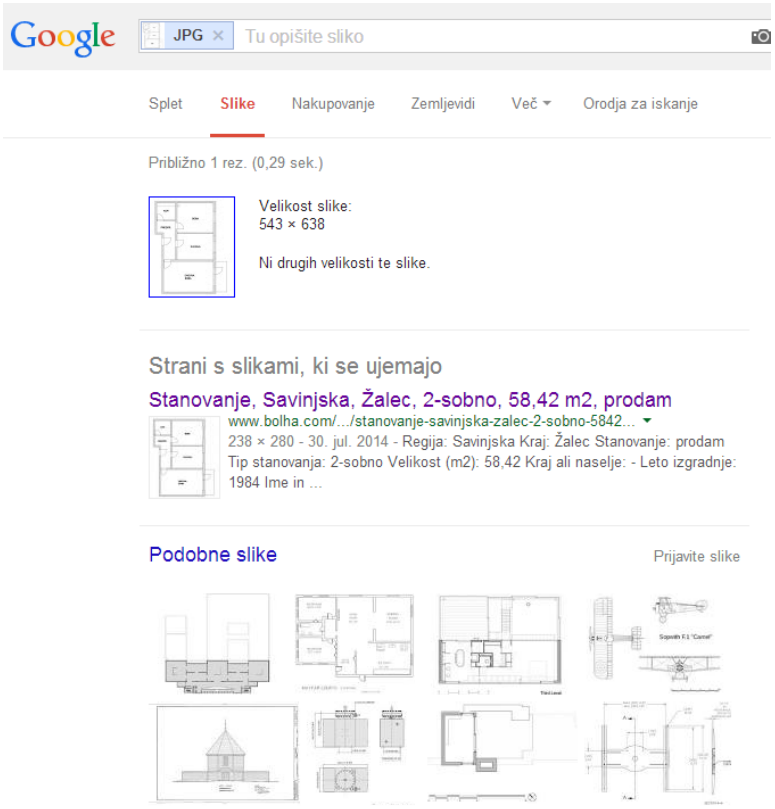
Če podrobneje pogledamo sliko (slika 10) na najdeni spletni strani, ugotovimo da pravzaprav ne gre za isto sliko.

Slika 10: Napačen rezultat iskanja z Googlom

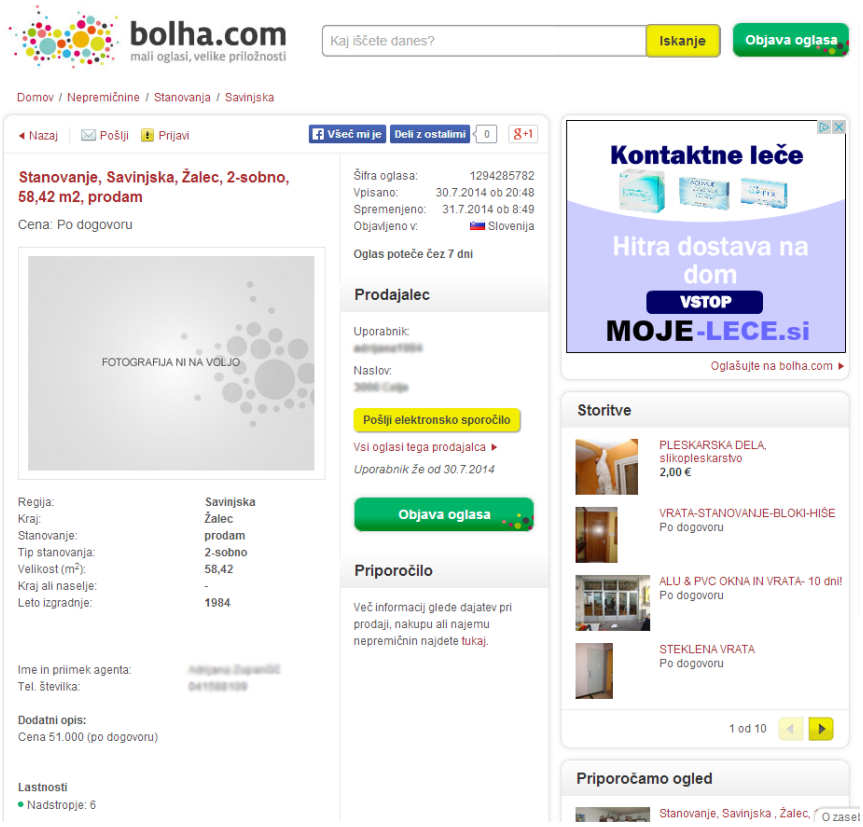
Občasno pa se dogaja, da nam Google kot zadetek vrne spletno stran, ki sploh ne vsebuje omenjene slike. To lahko vidimo v naslednjem primeru iz slike 11 in napačnih rezultatov na sliki 12 in 13:



Slika 11: Drugi primer slike za iskanje



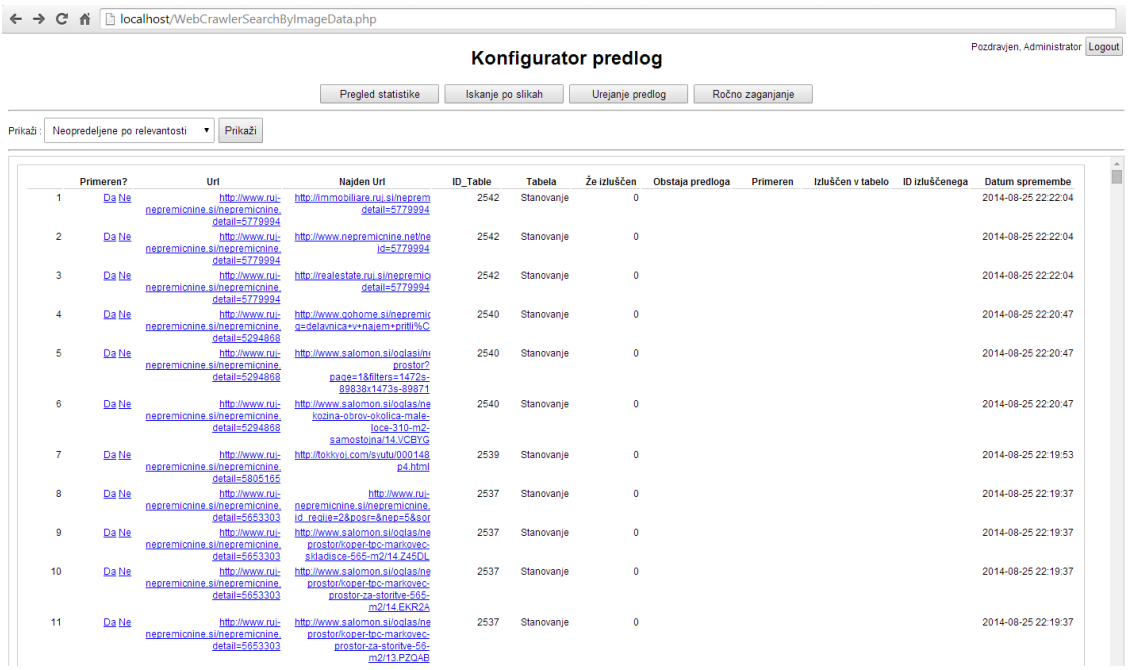
Slika 12: Drugi napačen rezultat iskanja 1/2



Slika 13: Drugi napačen rezultat iskanja 2/2

Kot je razvidno iz zgornjih dveh primerov, se ne moremo povsem zanašati na rezultate, ki nam jih vrne Google. Torej je še vedno potrebna fizična kontrola, da se ugotovi ali gre dejansko za isto sliko. Zaradi tega smo tudi v naši aplikaciji ročno preverjali ali so vsi najdeni zapisi primerni ali ne. Ta aplikacija sicer lušči samo tiste oglase za katere smo nastavili predloge.

Pri pripravi podatkov za analizo smo v našem primeru naleteli na še dodatno težavo, zaradi dveh vrst oglasov o nepremičninah; istočasna prodaja in najem ene nepremičnine v dveh različnih oglasih. V izogib tovrstnim napakam smo v našo aplikacijo dodali še dodatno funkcionalnost s katero je mogoče pregledovati posamezne strani, pri katerih naj bi šlo za isti oglas. V kolikor temu ni tako, imamo tu možnost (slika 14), da zanikamo primernost in to povezavo med njima onemogočimo.



Slika 14: Slikovno iskanje

3.2 Struktura predloge za luščenje

Obstaja kar nekaj standardov predlog za luščenje (ang. *Website Parse Template*), katere se razlikujejo v ontologijah. RDF, OWL in ICDL so nekatere izmed bolj priljubljenih ontologij. Izbrali bi lahko katerokoli, saj so vse enako zmogljive in za nas primerne, vendar smo za našo aplikacijo izbrali ICDL, ker je izredno berljiva in enostavna za razumevaje.

Predloga za luščenje vsebuje naslednje sklope [26]:

- **Ontologijo**, kjer so definirani koncepti in relacije, ki se uporabljajo na spletni strani.
- **Predloge -** Izdajatelj določa predloge za skupine spletnih strani, ki so si podobne po kategoriji in predvsem strukturi. Znotraj predloge je za vsak element HTML, ki ga želimo izluščiti, definirana pot XPath oziroma TagId.
- **URL** - Določeni dovoljeni vzorci URL, ki so primerni za skupino spletnih strani že opisanih v prejšnji točki.

Kot primer si poglejmo OWL in ICDL predlogo za luščenje:

OWL predloga [25]:

```
<?xml version="1.0" encoding="UTF-8"?>
<ow:wpt xmlns:ow="http://www.omfica.org/schemas/ow/0.9"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        ow:host="http://music.yahoo.com">

        <ow:ontology ow:name="general">
                <ow:concept ow:name="artist">
                        <ow:inherit>person</ow:inherit>
                        <ow:has>name</ow:has>
                        <ow:has>album</ow:has>
                        <ow:has>track</ow:has>
                        <ow:has>image</ow:has>
                        <ow:has>bio</ow:has>
                        <ow:has>video</ow:has>
                        <ow:has>id</ow:has>
                </ow:concept>
                <ow:concept>logo</ow:concept>
                <ow:concept>menu</ow:concept>
                <ow:concept>advertisement</ow:concept>
        </ow:ontology>

        <ow:template ow:name="Artist Page on Yahoo! Music"
                     ow:url="http://music.yahoo.com/ar-(artist.id[[0-9]*])---(artist.name[[A-Z,a-z,-,0-
9]*])"
                     ow:ontology="general">
                <ow:block ow:tagid=="yent-uhdr">Menu</ow:block>
                <ow:block ow:xpath="/html/body/div[2]/div/div/div[3]/div/a/span">Logo</ow:block>
                <ow:block ow:xpath="/html/body/div/div">Advertisement</ow:block>
                <ow:block
ow:xpath="/html/body/div[3]/table/tbody/tr/td[2]/div/h1">artist.name</ow:block>
                <ow:block ow:tagid="art_img">artist.image</ow:block>
                <ow:block ow:tagid="biography">artist.bio</ow:block>
                <ow:block
ow:xpath="/html/body/div[3]/table/tbody/tr/td[2]/table/tbody/tr[22]">artist.album</ow:block>
                <ow:block
ow:xpath="/html/body/div[3]/table/tbody/tr/td[2]/table/tbody/tr[10]">artist.track</ow:block>
                <ow:block
ow:xpath="/html/body/div[3]/table/tbody/tr/td[2]/table/tbody/tr[13]">artist.video</ow:block>
        </ow:template>

        <ow:urls ow:name="Artist page on Yahoo! Music" ow:template="Artist page on Yahoo! Music">
                <ow:url>http://music.yahoo.com/ar-8206256---Amy-Winehouse</ow:url>
                <ow:url>http://music.yahoo.com/ar-([artist.id[0-9]*])---(artist.name[[A-Za-z0-9-
]*])</ow:url>
        </ow:urls>
</ow:wpt>
```

ICDL predloga:

```
<?xml version="1.0" encoding="UTF-8"?>
<icdl host="http://music.yahoo.com">

        <ontology name="general" language="icdl:ontology">
             <concept name="Qbye Music">
                  <inherit concept="person"></inherit>
                  <has object="name"></has>
                  <has object="album"></has>
                  <has object="track"></has>
                  <has object="image"></has>
                  <has object="bio"></has>
                  <has object="video"></has>
                  <has object="id"></has>
                  <has object="fullname"></has>
             </concept>
             <concept name="Logo"></concept>
             <concept name="Menu"></concept>
             <concept name="Advertisement"> </concept>
        </ontology>

        <template name="Artist page on Yahoo! Music" language="icdl:template">
             <html_tag tagid="yent-uhdr" content="Menu"/>
             <html_tag xpath="/html/body/div[2]/div/div/div[3]/div/a/span" content="Logo"/>
             <html_tag xpath="/html/body/div/div" content="Advertisement"/>
             <html_tag xpath="/html/body/div[3]/table/tbody/tr/td[2]/div/h1" content="artist.name"/>
             <html_tag tagid="art_img" content="artist.image"/>
             <html_tag tagid="biography" content="artist.bio" reference="Artist Bio"/>
             <html_tag xpath="/html/body/div[3]/table/tbody/tr/td[2]/table/tbody/tr[22]"
content="artist.album"/>
             <html_tag xpath="/html/body/div[3]/table/tbody/tr/td[2]/table/tbody/tr[10]"
content="artist.track"/>
             <html_tag xpath="/html/body/div[3]/table/tbody/tr/td[2]/table/tbody/tr[13]"
content="artist.video"/>
        </template>

        <urls name="Artist page on Yahoo! Music" template="Artist page on Yahoo! Music">
             <url url="http://music.yahoo.com/ar-8206256---Amy-Winehouse"/>
             <url url="http://music.yahoo.com/ar-([artist.id[0-9]*])---(artist.full name[[A-Z,a-z,-,0-
9]*])"/>
        </urls>
</icdl>
```

Kot že omenjeno, sta obe predlogi enako zmogljivi, vendar je ICDL lažje berljiva. Za našo aplikacijo je v ontologiji opisana struktura objekta, ki ga bomo shranili v podatkovno bazo. V atributu *name* znotraj sekcije *concept* je zabeleženo ime tabele v podatkovni bazi, kamor se morajo shraniti vsi njegovi zapisi. V kolikor je potrebno iz določene spletne strani shranjevati v več tabel hkrati, je potrebno za vsako tabelo imeti svoj element *concept*. Znotraj elementa *concept* so elementi *has*, ki v našem primeru predstavljajo stolpec v tabeli.

Ker smo želeli, da bi naša aplikacija znala generično, ne glede na predlogo in strukturo podatkovne baze, luščiti in shranjevati v podatkovno bazo, smo v predlogo dodali še nekaj dodatnih atributov. V element *concept* smo dodali dva nova atributa *isMain* in *isVirtual*. Oba atributa imata za zalogo vrednosti *true, false*, 1 ali 0. V kolikor atributa nista nastavljena je njuna privzeta vrednost *false*. *IsMain* atribut predstavlja glavno tabelo, vse ostale so njej podrejene.

```
<concept name="Stanovanje" isMain="true">
```

IsVirtual pa predstavlja navidezno tabelo, katera se ne shranjuje v bazo, je pa včasih potrebna za različna preračunavanja kot na primer pretvorbo valut ali podobno.

```
<concept name="virtual" isVirtual="true">
  <has object="PriceNew" type="doubleSlo"></has>
</concept>
```

V element *has* smo dodali nov atribut *type*, ki pove kakšnega tipa je element luščen iz spletne strani. Na spletni strani je vse zapisano kot tekst, zato smo potrebovali dodatno informacijo, da lahko v kodi pretvorimo v pravilen tip in nato shranimo v podatkovno bazo. V atribut zapišemo eno od naslednjih vrednosti: *text, url, img, image, decimal, decimalSlo, double, doubleSlo, datetime*. V kolikor element nima nastavljenega atributa, je njegova privzeta vrednost *text*. Pri vrednostih *img* in *image* gre za isto vrednost in med njima ne razlikujemo. Dodana je bila samo zaradi morebitne uporabnikove zmotljivosti. Razlika med *decimal decimalSlo* je v decimalni vejici. V slovenski verziji je namreč vejica namesto pike. Enako velja za *double* in *doubleSlo*.

```
<has object="Price" type="doubleSlo"></has>
<has object="Image1" type="Image"   ></has>
```

V drugem sklopu predloge je za vsak stolpec tabele določeno, kje na spletni strani se nahaja podatek. To lahko določimo na dva načina; z določanjem poti do elementa *xpath* ali imenom elementa *tagid*. Poleg teh dveh možnosti pa smo dodali še dve dodatni opciji: *value* in *classname*. Za prvo opcijo je značilno, da ima fiksno vrednost, ki se jo nastavi že v predlogi in se je ne lušči iz spletne strani. Kot primer uporabe zgolj spletnih strani v Sloveniji, bo imela naša predloga nastavljeno fiksno vrednost »Slovenija«.

```
<html_tag value="Slovenia" content="Stanovanje.Country"/>
```

Druga dodana opcija deluje podobno kot *xpath* z razliko, da atribut *classname* poišče pojavitev elementa, ki ima nastavljen določen stil. V kolikor v dokumentu HTML obstaja več elementov, ki imajo nastavljen enak stil, se bo upoštevala prva pojavitev.

```
<html_tag classname="cena" content="Stanovanje.Price" />
```

Da bi se izognili spreminjanju kode aplikacije ob vsaki novi dodani predlogi, smo v element *html_tag* dodali še tri dodatne atribute: *date_format, regex* in *code*. Atribut *date_format* se uporablja v kombinaciji s prej omenjenim atributom *type*, kadar ima ta nastavljeno vrednost na *datetime*. Atributu *date_format* se nastavi format datuma, ki ga beremo iz spletne strani.

```
<html_tag value="1.3.2010" content="Stanovanje.Datum" date_format="d.m.Y" />
```

V atribut *regex* vpišemo regularni izraz, ki nam bo iz teksta izluščil samo želeno vsebino.

```
<html_tag classname="cena" content="Stanovanje.Price" regex="([0-9\.\,]+)[ ]*[€$]"/>
```

Atribut *code* pa nam omogoča, da vanj zapišemo kodo PHP, ki naj se izvrši nad objektom. Pri tem je potrebno upoštevati, da se vrednost izluščenega podatka poda v spremenljivko *[$valueIn]* in rezultat kode nastavi v spremenljivko *[$valueOut]*.

```
<html_tag xpath="/div[class=main-data]/td[6]" content="Stanovanje.SaleType" code="if ([$valueIn]
                    =='prodaja') [$valueOut] = 1 else [$valueOut] = 2" />
```

V zadnjem sklopu predloge so podane povezave URL, ki so primerne za luščenje podatkov s trenutno predlogo. Povezave URL so lahko zapisane tudi s pomočjo regularnega izraza, kar pa uporabimo predvsem takrat kadar so določene strani generirane na podlagi iste skripte in imajo posledično enako strukturo in zelo podoben URL naslov.

```
<urls>
  <url url="http\:\/\/www.nepremicnine.net\/nepremicninE.html\?id\=[0-9]*"/>
</urls>
```

3.3 Struktura podatkovne baze

Kot je že bilo omenjeno smo izbrali podatkovno bazo MySql, z razlogom ker je brezplačna in povsem zadovoljiva za naše potrebe. Zasnovana je trinivojsko. Na prvem nivoju so tabele, ki so potrebne za luščenje podatkov. V njih so shranjene nastavitve in statistike. Na drugem nivoju so tabele, ki vsebujejo izluščene podatke. Na zadnjem nivoju pa tabele, ki hranijo spremembe izluščenih podatkov skozi čas.

Pomembno vlogo v naši aplikaciji imajo sprožilci (ang. *trigger*), ki se avtomatično prožijo ob dodajanju ali spremembah in avtomatično polnijo zgodovinske zapise v vse tabele na tretjem nivoju. Na podlagi te funkcionalnosti lahko, na primer, pregledujemo časovno vrsto cene nepremičnine.

Na prvem nivoju so naslednje tabele:

- WptSettings
- WebParseTemplate
- WebParseTemplate_History
- WebParseTemplateItems
- WebCrawlerSchedule
- WptSearchByImage

Na drugem nivoju so poljubne tabele. Njihovo ime in imena njenih stolpcev niso pomembna saj so konfigurabilna s predlogo za luščenje. Edina zahteva je, da morajo vsebovati stolpce URL, *DateModified* in *Active*. V stolpcu URL je shranjen podatek od kod se je vsebina celotne vrstice izluščila, v *DateModified* je podatek o datumu in času, ko se je vsebina izluščila in *Active* vsebuje podatek o aktualnosti vsebine. V primeru, da spletna stran, ki je podana s stolpcem URL, ni več dosegljiva, lahko predvidevamo, da je bil oglas izbrisan. S

pomočjo *Active* stolpca si zagotovimo mehko brisanje. Sicer zapise fizično ne brišemo, da imamo možnost vpogleda tudi v stare zapise.

Vsaka tabela iz drugega nivoja mora imeti enako imensko tabelo na tretjem nivoju pri čemer ima na koncu imena dodana še končnica *_History*. Prav tako mora vsebovati enake stolpce kot jih vsebuje tabela na drugem nivoju.

3.3.1 Tabela »WptSettings«

Tabela *WptSettings* vsebuje nastavitve aplikacije ter vsebuje naslednje stolpce:

- *Section* – hrani podatek o pripadnosti sekciji nastavitev (npr. *Common*, *Image*, *DataBase* …)
- *Name* – ime nastavitve
- *Value* – vrednost nastavitve
- *IsSettings* – podatek, ki pove ali gre za nastavitev aplikacije ali le za shranjeno nastavitev, kar vpliva na delovanje aplikacije. Nekatere podatke je namreč potrebno shranjevati in se le-ti med izvajanjem spreminjajo. Na primer; za slikovno iskanje se shranjuje čas zadnjega iskanja, s tem pa dosežemo, da ne obremenjujemo preveč strežnika na katerega pošiljamo poizvedbe.
- *Comment* – komentar za posamezno nastavitev

3.3.2 Tabela »WebParseTemplate«

Tabela *WebParseTemplate* ima shranjene predloge za luščenje ter vsebuje naslednje stolpce:

- *ID_WebParseTemplate* – identifikacijska številka predloge za luščenje
- *Name* – naziv predloge
- *XmlTemplate* –vsebina predloge XML
- *DateModified* – čas in datum zadnje spremembe
- *Active* – veljavnost zapisa. V kolikor je bil zapis izbrisan se ga fizično ne odstrani ampak se mu nastavi Active = 0

3.3.3 Tabela »WebParseTemplate_History«

Tabela *WebParseTemplate_History* shranjuje zgodovinske zapise tabele *WebParseTemplate*. Polnjenje se izvaja avtomatsko s pomočjo sprožilcev:

1. Za novo dodane zapise

```
CREATE TRIGGER `WebParseTemplateHistoryInsertTrigger` AFTER INSERT ON `
WebParseTemplate`
 FOR EACH ROW BEGIN

INSERT INTO `WebParseTemplate_History` (ID_WebParseTemplate, Name, XmlTemplate, DateModified,
Active) VALUES
(NEW.ID_WebParseTemplate, NEW.Name, NEW.XmlTemplate, NEW.DateModified, NEW.Active);

END
```

2. Za spremenjene zapise

```
CREATE TRIGGER `WebParseTemplateHistoryUpdateTrigger` AFTER UPDATE ON ` WebParseTemplate`
 FOR EACH ROW BEGIN

INSERT INTO `WebParseTemplate_History` (ID_WebParseTemplate, Name, XmlTemplate, DateModified,
Active) VALUES
(NEW.ID_WebParseTemplate, NEW.Name, NEW.XmlTemplate, NEW.DateModified, NEW.Active);

END
```

3.3.4 Tabela »WebParseTemplateItems«

Tabela *WebParseTemplateItems* hrani podatke za povezavo med predlogo in izluščenim objektom/stranjo. *WebParseTemplateItems* vsebuje naslednje stolpce:

- *ID_WebParseTemplate* – identifikacijska številka predloge za luščenje
- *ID_Table* – identifikacijska številka zapisa, ki je shranjena v tabeli *TableName*
- *TableName* – ime tabele v katero se je zapis shranil

Ta povezava je potrebna za kasnejšo pregledovanje s predlogo, če ima oglas morebitne spremembe ali le-ta ne obstaja več.

3.3.5 Tabela »WebCrawlerSchedule«

Tabela *WebCrawlerSchedule* hrani statistiko posameznih luščenj in je v pomoč v primerih, kadar imamo nastavljen določen časovni razmik med posameznimi iskanji z isto. *WebCrawlerSchedule* vsebuje naslednje stolpce:

- *ID_WebCrawlerSchedule*– identifikacijska številka
- *ID_WebParseTemplate* – identifikacijska številka predloge za luščenje
- *DateStart* – Datum in čas začetka preiskovanja
- DateEnd – datum in čas konca preiskovanja
- NumOfCrawledPages – število strani, ki so ustrezale predlogi

3.3.6 Tabela »WptSearchByImage«

Tabela *WptSearchByImage* vsebuje rezultate poizvedb, ki jih vrne Googlovo slikovno iskanje. Podatki nam omogočajo kasnejšo preiskovanje strani ujemajočih oz. podobnih slik. S to tabelo lahko ugotavljamo povezavo med posameznimi oglasi, kar ji daje dodatno uporabno vrednost. *WptSearchByImage* vsebuje naslednje stolpce:

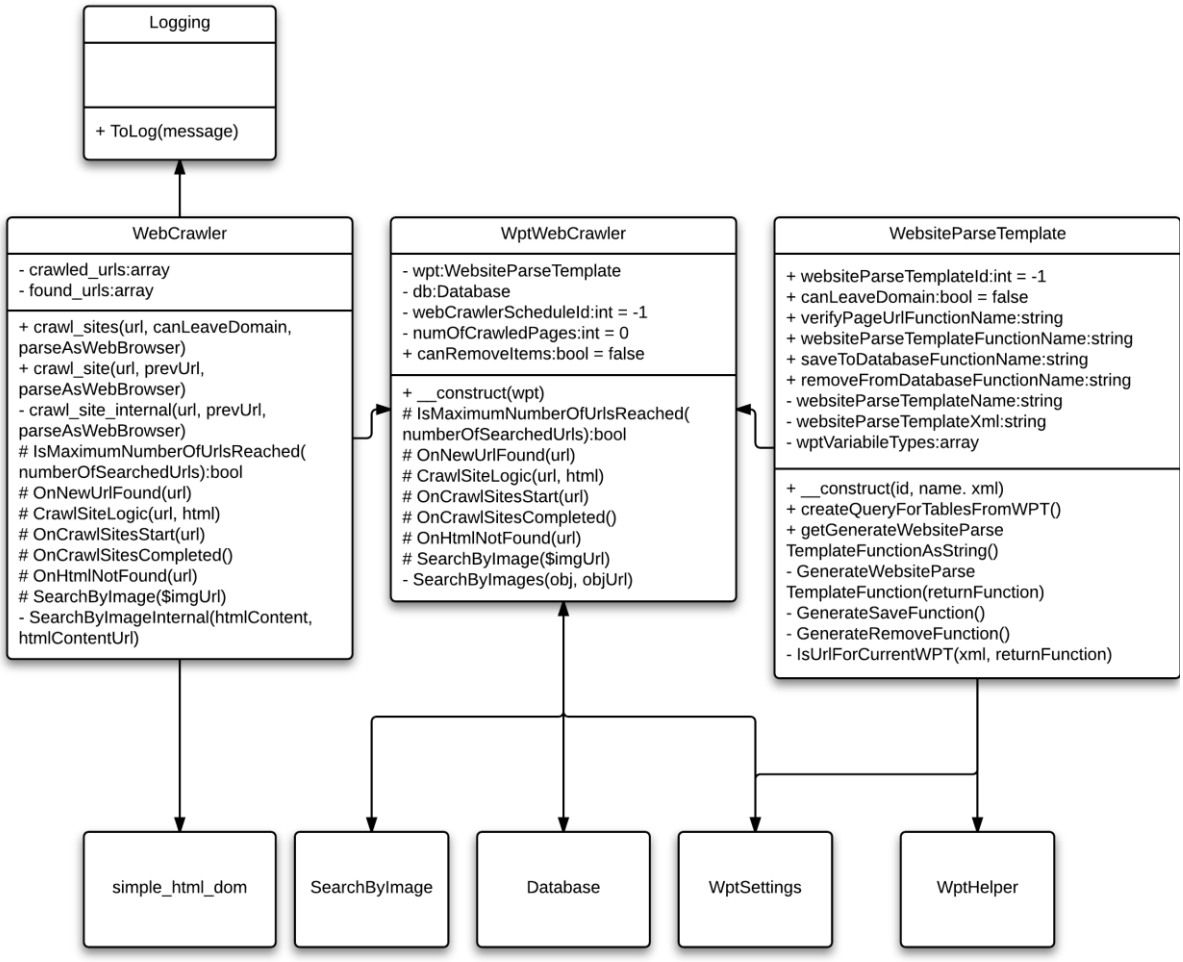
- *TableName* – Ime tabele, kjer se nahaja zapis za katerega smo prožili slikovno iskanje
- *ID_Table* – Identifikacijska številka zapisa za katerega smo prožili slikovno iskanje
- *UrlWithSameImage* – Naslov URL, kjer je bila najdena enaka slika
- *IsCrawled* – Podatek ali je bil naslov URL že preiskan
- *ExistWptTemplate* – Podatek ali za omenjen naslov URL obstaja predloga za luščenje
- *IsRelevant* – Podatek ali je povezava URL, ki jo je vrnil Google res relevantna za naš oglas. Včasih se zgodi da Google vrne tudi strani, ki si niso podobne. Ta podatek ni mogoče pridobiti avtomatsko, ker je kar težko ugotoviti ali gre za primerno informacijo ali ne
- *CrawledToTableName* – Ime tabele kamor se je shranil zapis, ki je bil preiskan s pomočjo naslova URL *UrlWithSameImage*
- *CrawledToID_Table* – identifikacijska številka zapisa, ki je bil preiskan s pomočjo naslova URL *UrlWithSameImage*

3.4 Arhitektura rešitve

Bistveno ogrodje aplikacije je predstavljeno s spodnjim diagramom (slika 15). *WebCrawler.php* je abstraktni razred, ki vsebuje osnovno logiko za plezanje po spletnih straneh, več od tega pa ne zmore. Osnovan je bil na skripti, ki jo je spisal Subin Siby in jo lahko najdemo na http://subinsb.com/how-to-create-a-simple-web-crawler-in-php. *WptWebCrawler.php* je dedovan iz *WebCrawler.php* ter ima poleg dedovane logike, še nekaj dodatne specifične logike, ki smo jo uporabili za našo aplikacijo (preštevanje strani, beleženje preiskanih strani, slikovno iskanje …). Celotno logiko luščenja podatkov iz določene strani vsebuje *WebsiteParseTemplate.php*, ki deluje povsem generično glede na podano predlogo.

Slika 15: Bistveno ogrodje aplikacije

Bistvene sestavne dele aplikacijo in povezavo med njimi prikazuje spodnji diagram (slika 16). Osrčje aplikacije je WptWebCrawler.php. Bolj podroben opis posameznih delov sledi v naslednjih poglavjih.



Slika 16: Bistveni deli aplikacije

3.4.1 WebCrawler.php

Kot rečeno abstraktni razred *WebCrawler.php* vsebuje osnovno logiko za plezanje po spletnih straneh. S pomočjo virtualnih funkcij lahko skripte, ki to logiko dedujejo, določajo dodatno logiko, kako postopati s posamezno od najdenih strani. Aplikacija s pomočjo skripte *Logging.php*, morebitne napake zapiše v dnevno log datoteko. Plezanje po spletnih straneh se izvaja s pomočjo knjižnice *simple_html_dom.php*, s katero iščemo URL naslove na spletni strani, ki jih bo potrebno preiskati. V nadaljevanju pa bomo z njo tudi luščili podatke. *WebCrawler.php* podpira dve vrsti plezanja:

1. Plezanje po spletni strani, za katero podamo naslov URL
2. Plezanje po spletni strani, za katero podamo naslov URL in vseh spletnih straneh, ki imajo povezavo na tej spletni strani

V obeh primerih mora skripta, ki je dedovana iz naše skripte redefinirati nekatere od naslednjih funkcij:

- *IsMaximumNumberOfUrlsReached* – Če je dovoljeno plezanje v globino, ker še ni bila prekoračena globinska širina, funkcija vrača true, sicer false.
- *OnNewUrlFound* – Funkcija, ki vsebuje logiko, ki jo je potrebno izvršiti, ko je najden nov naslov URL spletne strani. Vhodni parameter funkcije je novo najdeni URL.
- *CrawlSiteLogic* – Funkcija z logiko, ki jo je potrebno izvršiti, ko je preiskana nova spletna stran. Vhodni parameter funkcije je URL najdene spletne strani in njena vsebina. Vsebina spletne strani je pridobljena s pomočjo knjižnice simple_html_dom.php.
- *OnCrawlSitesStart* – Funkcija, ki vsebuje logiko, katero je potrebno izvršiti, ko se prične plezanje. Vhodni parameter funkcije je začetni naslov URL.
- *OnCrawlSitesCompleted* – Funkcija, ki vsebuje logiko, ki jo je potrebno izvršiti po zaključku plezanja.
- *OnHtmlNotFound* – Funkcija, ki vsebuje logiko, ki jo je potrebno izvršiti, ko spletna stran na podanem naslovu URL ne obstaja. Vhodni parameter funkcije je naslov URL, ki ne obstaja.

3.4.2 WebSiteParseTemplate.php

WebSiteParseTemplate vsebuje predlogo za luščenje, ki je bila opisana v poglavju 2.2. Na podlagi predloge generira še naslednje funkcije:

- Funkcija za generiranje tabel in prožilcev za podatkovno bazo MySql

- Funkcija za preverjanje ali je format URL ustrezen (določen v predlogi znotraj sklopa URLS)
- Funkcijo za luščenje vsebine oglasa iz podane spletne strani
- Funkcija za shranjevanje izluščenih podatkov oglasov v podatkovno bazo MySql
- Funkcija za deaktiviranje oglasa, v kolikor ne obstaja več

Vse omenjene funkcije se sestavljajo generično v kodo PHP kot tekst, ki se nato z vgrajeno funkcijo PHP ***create_function*** pretvori v funkcijo. Kot primer bi predstavili najkrajšo in najenostavnejšo izmed omenjenih, to je funkcija za preverjanje ustreznosti formata URL določenega v predlogi:

```
$condition = '';

//Sprehodi se čez vse dovoljene url-je
foreach($xml->urls->url as $allowedurl)
{
  //Dodaj vsako nadaljne regex preverjanje z OR pogojem
  $condition .= ($condition == '') ? '' : ' || ';

  //Dodaj regex pogoj
  $condition .= 'preg_match("/'.$allowedurl['url'].'/i", $url)';
}


$functionContentAsString = ($condition != '') ?
              'if('.$condition.') return true; else return false;' :
              'return true;';

$funcname = create_function(
              '$url',                    /*argumenti funkcije*/
              $functionContentAsString   /*funkcija podana kot teskst*/
              );

return $funcname;
```

3.4.3 SearchByImage.php

Obstaja kar nekaj iskalnikov, ki podpirajo slikovno iskanje. Vendar pa noben od ponudnikov ne ponuja API-ja z omenjeno funkcionalnostjo. Google sicer nudi API za iskanje, s katerim pa je mogoče le tekstovno iskanje. Zaradi tega smo se naloge v začetku lotili s preučevanjem načina kako Google generira povezavo URL ob iskanju po sliki. Izkazalo se je, da je generiranje povezav zelo preprosto, če gre za sliko s spleta. V tem primeru je potrebno v parameter podati le URL slike:

http://www.google.com/searchbyimage?hl=sl&image_url={url_slike}

Ko smo to logiko sprogramirali, pa se je izkazalo, da bo do rezultatov potrebno dostopati nekoliko drugače. Google namreč preverja način dostopa in ga ne dovoljuje spletnim pajkom. Zato je bilo potrebno našo skripto predelati, da se je odzivala kot, da ne gre za avtomatično iskanje. To smo naredili s pomočjo knjižnice cURL, ki ponuja več kontrole nad prenosom. S pomočjo knjižnice lahko pošiljamo zahtevke na enak način kot bi to počel internetni brskalnik, ki ga uporablja uporabnik. To smo dosegli z nastavitvijo dodatnih parametrov. Najpomembnejši za naš primer so:

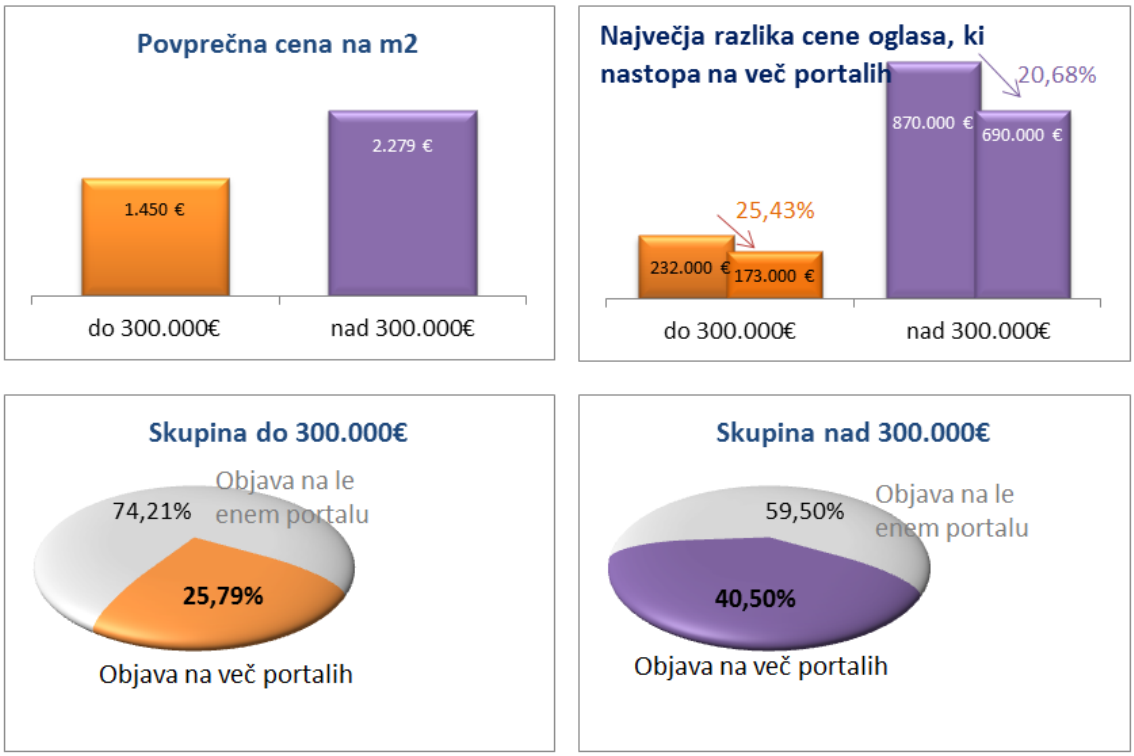
- CURLOPT_URL – URL, ki ga želimo obiskati.
- CURLOPT_USERAGENT – Možnost identifikacije kot običajni brskalnik.
- CURLOPT_RETURNTRANSFER – Ali želimo rezultat shraniti v spremenljivko.
- CURLOPT_COOKIEFILE – Naslov, kjer se nahaja piškotek.
- CURLOPT_COOKIEJAR – Naslov, kjer se nahaja piškotek.
- CURLOPT_FOLLOWLOCATION – Ali naj se preusmeritve na strani upoštevajo.
- CURLOPT_REFERER – Naslov URL iz katerega prihajamo.

4 Analiza

S pomočjo aplikacije smo izluščili skoraj 2.500 nepremičninskih oglasov iz različnih portalov. Oglase smo razvrstili v dve skupini, na cenejše oz. dražje od 300.000 €. Na podlagi izluščenih oglasov in omenjenega rangiranja smo pridobili zanimive podatke, ki pa bi bili za kupca uporabne informacije. Za namen prikaza mogočih informacij pridobljenih s to aplikacijo, podajamo spodnji pregled in sliko 17.

Povprečna cena cenejših nepremičnin je 121.492 €, medtem ko je povprečna cena dražjih 1.188.882 €. Povprečna cena kvadratnega metra pri cenejših je 1.450€ in 2.279 € pri dražjih. Nekateri izmed prodajalcev se odločajo, da svojo nepremičnino prodajajo na več portalih hkrati. Tako se na različnih portalih pojavlja 25,8 % cenejših oglasov in kar 40,5 % je takšnih med dražjimi. Izmed teh, ki so na različnih portalih, je 15 % prodajalcev postavilo tudi različne cene. Cene v nižji skupini oglasov med posameznimi portali v povprečju odstopajo za 2,3 %, pri dražjih pa za 1,9 %. Maksimalna razlika v ceni prve skupine je 25,4 %, druge pa 20,7 %.

Hkrati smo za vsak oglas spremljali tudi njegovo spreminjanje cene v časovnem obdobju; pri 13,0 % oglasov se cena skozi čas spreminja. V poprečju cena nepremičnine v času prodaja pade za 11,4 %. Maksimalni padec cene, ki je bil zabeležen, je znašal kar 45,5 %.



Slika 17: Grafični prikaz iz analize podatkov

5 Zaključek in nadaljnje delo

V diplomski nalogi je bil narejen prototip aplikacije za primerjavo cen nepremičnin na različnih nepremičninskih portalih. S tem prototipom je mogoče luščili podatke iz oglasov s kateregakoli nepremičninskega portala, hkrati pa tudi sam poveže iste oglase na različnih portalih. S tovrstno aplikacijo bi bilo mogoče analizirati nepremičninski trg na slovenskih portalih, s tem pa bi kupci nepremičnin lahko spremljali cene nepremičnin skozi čas ter našli najugodnejšo ceno za svoj nakup.

V prototip bi bilo mogoče dodati še nekatere izboljšave:

1. Funkcije, ki so generirane s pomočjo ***create_function*** so počasnejše. Vse funkcije v prototipu generirane na tak način se izvajajo med luščenjem podatkov. Za prototip ta hitrosti sicer ni problematična, če pa bi ga razvili v prodkucijsko aplikacijo, bi bilo smiselno razmisliti koliko bi lahko pridobili na času, če bi generirali fizično datoteko, ki bi vsebovala PHP kodo za vsako od predlog. Datoteke bi se generirale avtomatsko ob prvem shranjevanju in ob vsaki spremembi predloge. Skripta bi vsebovala samo funkcije, ki se trenutno genrirajo s pomočjo ***create_function***.
2. Trenutno naša aplikacija ne omogoča avtomatskega izločanja napačno poiskanih slik in jih je potrebno ročno izločiti. Če bi aplikacija najdeni sliki primerjala po pikslih, bi lahko ugotavljali ali gre res za isti sliki.
3. Prevajanje v slovenščino – V kolikor luščimo podatke iz tujih portalov, bi lahko s pomočjo Google prevajalnika oglase prevajali v slovenski jezik.

Viri in literatura

[1] Victoria Shannon , "A 'more revolutionary' Web", International Herald Tribune. 2006

[2] Allan M. Collins, Allan M., Elizabeth F. Loftus. "A spreading-activation theory of semantic processing", Psychological Review, str: 407–428, 1975.

[3] (2013) Clever Uses for Reverse Image Search. Dostopno na: http://lifehacker.com/clever-uses-for-reverse-image-search-473032092

[4] (2014) Data scraping. Dostopno na: http://en.wikipedia.org/wiki/Data_scraping

[5] Allan M. Collins, M. Ross Quillian . "Does category size affect categorization time?", Journal of verbal learning and verbal behavior, str: 432–438, 1970.

[6] (2012) De Web 1.0 a Web 3.0. Dostopno na: http://www.koala-soft.com/de-web-10-a-web-30

[7] (2014) Image Search Engines: A Collection of the Best on the Web. Dostopno na: http://websearch.about.com/od/imagesearch/a/image.htm

[8] (2008) Kaj je spletni pajek – spider? Dostopno na: http://www.presentia.si/baza-znanja-helpdesk/2008/spletni-pajek-spider/

[9] (2014) Metadata. Dostopno na: http://en.wikipedia.org/wiki/Metadata

[10] (2012) Ontology. Dostopno na: http://semanticweb.org/wiki/Ontology

[11] (2005) Clay Shirky, Ontology is Overrated: Categories, Links, and Tags. Dostopno na: http://www.shirky.com/writings/ontology_overrated.html

[12] (2014) Optical character recognition. Dostopno na: http://en.wikipedia.org/wiki/Optical_character_recognition

[13] Allan M. Collins, M. R. Quillian. "Retrieval time from semantic memory", Journal of verbal learning and verbal behavior, str. 240–247, 1969.

[14] (2014) Reverse image search. Dostopno na: https://support.google.com/websearch/answer/1325808?hl=en

[15] (2014) ScrapeSentry Scraping Threat Report 2014. Dostopno na: http://www.scrapesentry.com/scrapesentry-scraping-threat-report-2014

[16] (2014) Screen Scraping. Dostopno na: http://www.techopedia.com/definition/16597/screen-scraping

[17] (2014) Semantic Web. Dostopno na: http://en.wikipedia.org/wiki/Semantic_Web

[18] (2014) Splet 2.0. Dostopno na: http://sl.wikipedia.org/wiki/Splet_2.0

[19] Peter Grlica, Tehnike spletnega luščenja podatkov, Diplomsko delo, 2013

[20] (2014) Ultimate Web Scraper Toolkit Documentation. Dostopno na: http://barebonescms.com/documentation/ultimate_web_scraper_toolkit/

[21] (2014) Vocabularies. Dostopno na: http://www.w3.org/standards/semanticweb/ontology

[22] (2011) Web 1.0 vs Web 2.0 vs Web 3.0 vs Web 4.0 – A bird's eye on the evolution and definition. Dostopno na: http://flatworldbusiness.wordpress.com/flat-education/previously/web-1-0-vs-web-2-0-vs-web-3-0-a-bird-eye-on-the-definition/

[23] (2007) Web 3.0 – Semantični splet. Dostopno na: http://blog.morphix.si/2007/uporabniske-izkusnje/web-30-%E2%80%93-semanticni-splet/

[24] (2014) Web scraping. Dostopno na: http://en.wikipedia.org/wiki/Web_scraping

[25] (2009) Website Parse Template. Dostopno na: http://www.w3.org/Submission/WPT/

[26] (2014) Website Parse Template. Dostopno na: http://en.wikipedia.org/wiki/Website_Parse_Template

[27] (2014) Tom Gruber, What is an Ontology? Dostopno na: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[28] Quillian, MR, "Word concepts. A theory and simulation of some basic semantic capabilities", Behavioral Science str: 410–430, 1967.